A Security Architecture for Large-Scale Distributed Computations

Ian Foster*   Carl Kesselman+

Gene Tsudik+   Steven Tuecke*

* Argonne National Laboratory

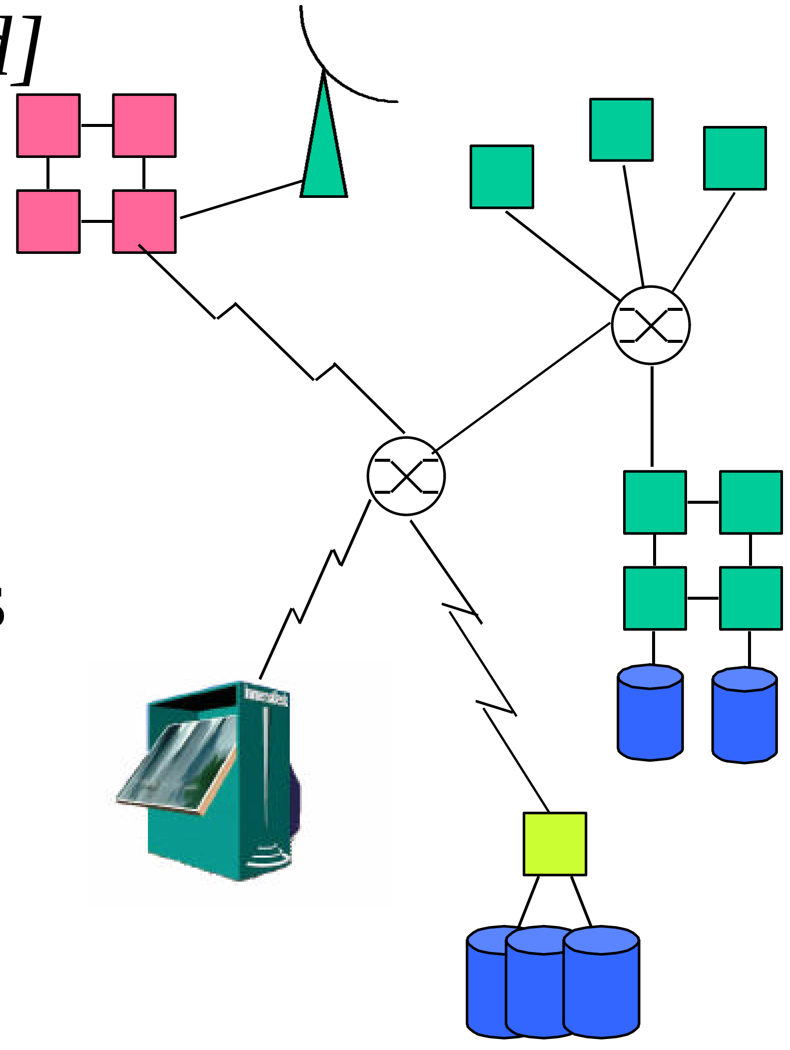
\+ USC Information Sciences Institute

Overview

- “Computational grids”
- The grid security problem
- Globus toolkit
- Globus security policy
- Globus security architecture
- Globus security implementation

Computational Grids

"Dependable, consistent, pervasive access to [high-end] computational resources"

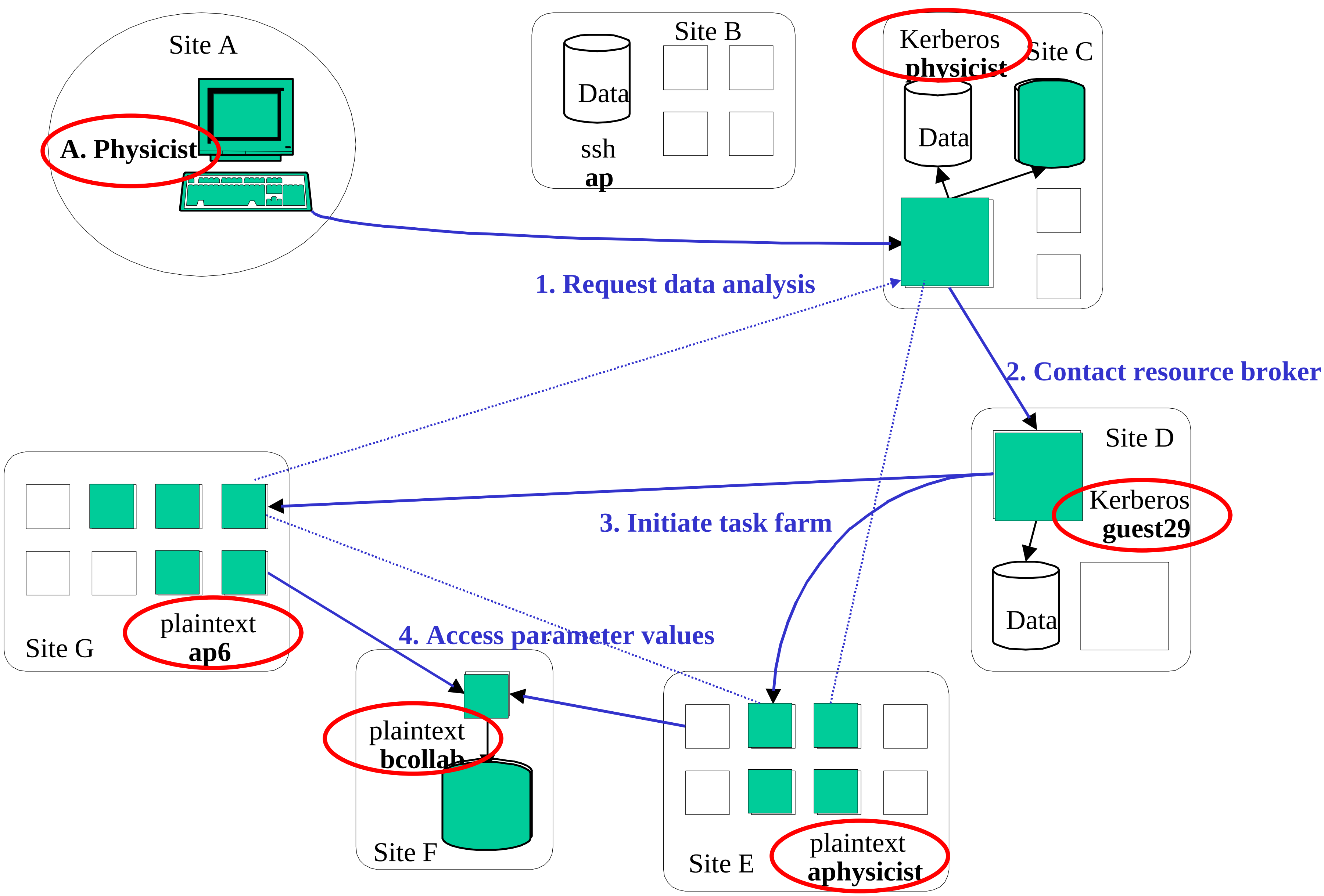


- Dependable: Can provide performance and functionality guarantees
- Consistent: Uniform interfaces to a wide variety of resources
- Pervasive: Ability to "plug in" from anywhere

Applications

- Remote visualization & computational steering
- Remote data analysis
- Online instruments
- Collaborative environments
- Distributed supercomputing

Motivating Example



Grid Security Characteristics

- Large & dynamic user population
- Large & dynamic resource population
- Computations grow/shrink dynamically
- Multiple communication methods
- Different local security solutions
- Different local credentials
- International users and resources

Requirements

- Single sign-on
- Protection of credentials
- Interoperability with local security solutions
- Exportability
- Uniform credentials/certification infrastructure
- Scalability
- Support for secure group communication
- Support for multiple implementations

Globus Toolkit

Chemical engineering
Cosmology
Environmental hydrology
Molecular biology
Scientific instrumentation
Nanomaterials
. . .

Numerical libraries
HPC++
Condor
Legion
MPI
Web tools
SWIG
CAVERN soft
Nimrod
Resource brokers
. . .
Many Worlds

Accounting
Communications
Information
Scheduling
Fault detection
Security
Instrumentation
QoS
Data access

Convex
SGI
SP
NT
LSF
Kerberos
. . .

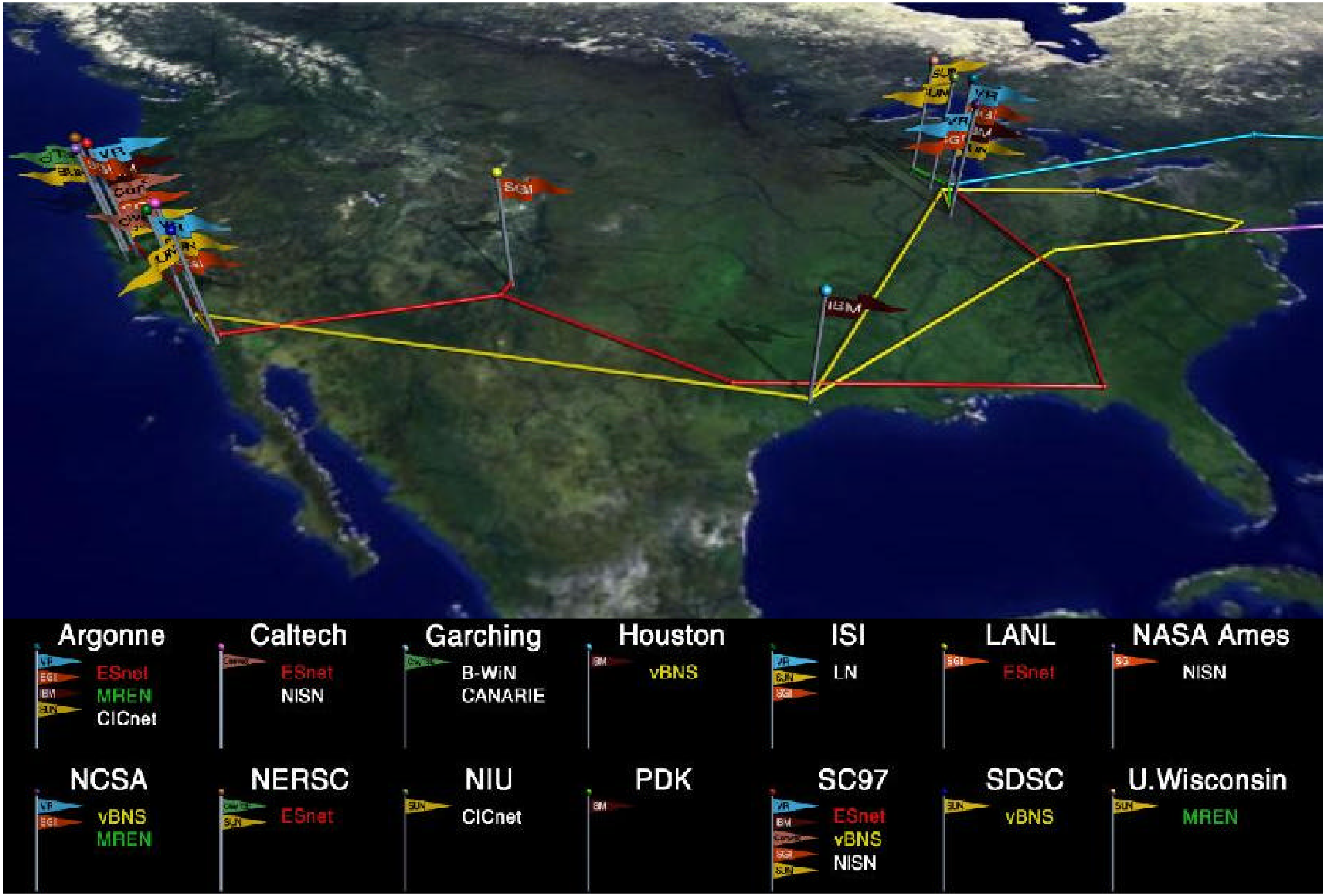
Core Globus Services

- Scheduling (Globus Resource Access Manager)
  - Low-level scheduler API
- Information (Metacomputing Directory Service)
  - Uniform access to structure/state information
- Communications (Nexus)
  - Multimethod communication + QoS management
- Security (Globus Security Infrastructure)
  - Single sign-on, key management
- Health and status (Heartbeat monitor)
- I/O (Global Access to Secondary Storage)
- Code (Globus Executable Management)

High-Level Services Include ...

- Communication libraries
  - MPICH, PAWS, etc.
- Parallel languages
  - CC++, HPC++
- Data access
  - RIO: remote I/O from MPI-IO
- Collaborative environments
  - ManyWorlds, CAVERNsoft
- Resource brokers
  - DUROC, Nimrod: high-performance, high-thruput

GUSTO Grid Testbed



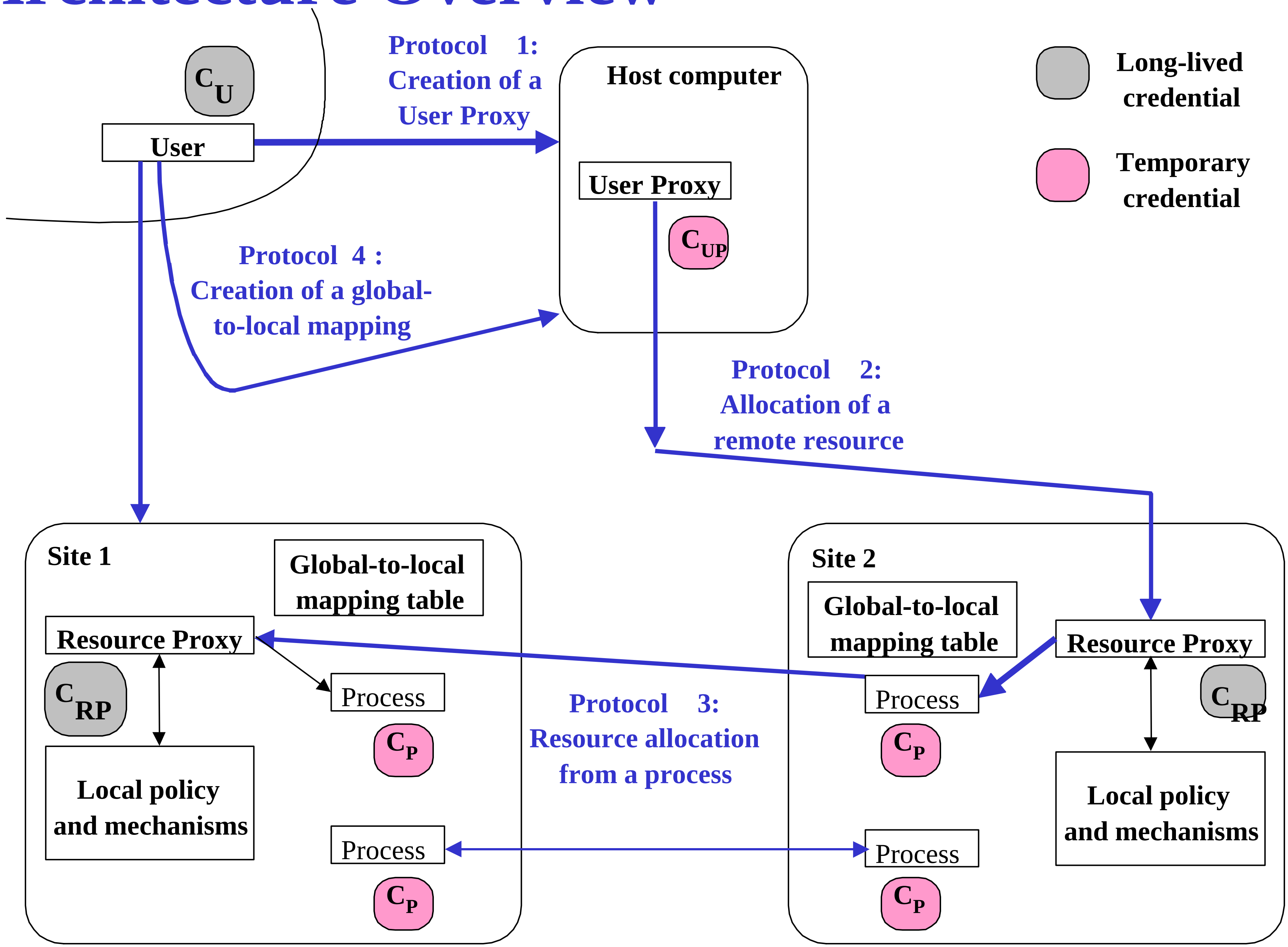
Approach: Policy

- Focus on security for interdomain operations
  - Assume that intradomain security is handled by existing mechanisms, which remain in place
- Allow for a program to act as a Globus user
  - Necessary for single sign-on when programs acquire resources dynamically
- Mutual authentication for interdomain operations
  - Site must accept validity of authentication

Policy Contd.

- Distinct global subject space and local subject space; former can be mapped to the latter
  - Mapping can be many-to-one, one-to-one, etc.
- Domain responsible for access control, etc., given local subject
  - Hence reuse of local mechanisms
- Processes running on the same resource for the same user can share credentials
  - Contributes to scalability

Architecture Overview



Architectural Components

- User proxy
  - Act as user for fixed time period
- Resource proxy
  - Translate between global and local security solns
- Mapping table
  - Translate global to resource subject

Protocols

1: Create user proxy (and temporary credential)

2: Allocate remote resource/create process(es)

– Mutual authentication, translation, access control

3: Resource allocation from user process

– Pass request from process to user proxy

4: Establish entry in mapping table

– Establish possession of global & local credentials

Protocols are mechanism independent and do not use encryption

1. User Proxy Creation

- User logs on to local host computer
- Produce user proxy credential:

  $C_{UP} = C_U$(user-id, host, start-time, end-time, ...)
- Create user proxy with C_{UP}

2. Resource Allocation

- User proxy locates appropriate resource proxy
- UP and RP authenticate using C_{UP} and C_{RP}
- UP presents RP with a signed request
- RP checks to see if the user who signed the UP's credentials is authorized by local policy to make the allocation request.

2b. Establish Process Credential

- RP creates tuple describing user, resource, etc.
- RP signs tuple and sends securely to the UP
- If UP wishes to approve the request, it signs the tuple to produce process credential C_P
- UP returns C_P securely to RP
- RP allocates resources, create process(es), passes C_P

3. Request from User Process

- Process and UP authenticate
- Process issues a request to UP

  C_P (“allocate”, allocation request parameters)
- UP executes request using Protocol 2
- The result is signed by the user proxy and returned to the requesting process.

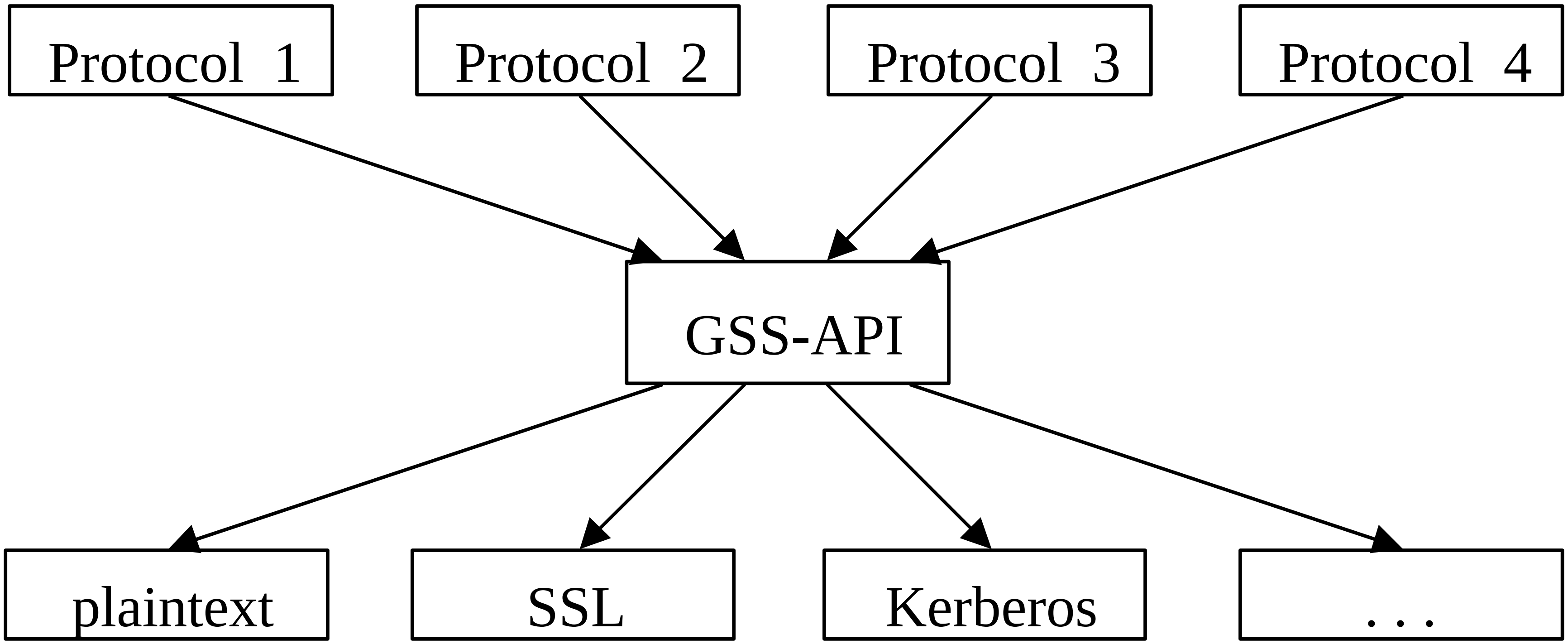
4. Establish Mapping Table Entry

- Basic idea is that user authenticates themselves both globally and locally, using global and local credentials; and then asserts that a mapping table entry should exist.

Implementation

- GSS-API provides mechanism-independence in inter-domain operations
  - Plaintext and public key implemented
  - Kerberos also possible
- PK based on SSL Authentication Protocols
- Multiple resource proxies implemented
  - "Globus Resource Allocation Manager"
  - Plaintext (rsh) + Kerberos implemented to date
- Deployed on heterogeneous testbed (20 sites)

Use of GSS-API



Evaluation

- ✓ Single sign-on
- ✓ Protection of credentials
- ✓ Interoperability with local security solutions
- ✓ Exportability
- ✓ Uniform credentials/certification infrastructure
- Scalability
- Support for secure group communication
- ✓ Support for multiple implementations

Scalability

- In number of processes created
  - Yes (?)
- In number of users and resources
  - Outstanding issue
- Group communication
  - Future work (e.g., CLIQUES)

Future Work

- Current architecture provides a solid basis for future work in several areas
  - More flexible policy-based access control
  - Inter-domain access control policies
  - Group communication
  - Scalability in number of users and resources
  - Flexible application of integrity and privacy mechanisms

Summary

- Distributed computing introduces unique security concerns
- Globus security architecture incorporates solutions to several key problems
- An implementation is operational at multiple sites and in use by applications
- A solid framework for further research and development